

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

INFORMATION PAGE

Form Approved
OMB No. 0704-0188

1a. AD-A214 635			1b. RESTRICTIVE MARKINGS		
2a. 1989			3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.		
2b. B			4. PERFORMING ORGANIZATION REPORT NUMBER(S)		
5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR-88-1308			6a. NAME OF PERFORMING ORGANIZATION University of Maryland		
6b. OFFICE SYMBOL (If applicable)			7a. NAME OF MONITORING ORGANIZATION Air Force Office of Scientific Research		
6c. ADDRESS (City, State, and ZIP Code) Department of Computer Science College Park, MD 20742			7b. ADDRESS (City, State, and ZIP Code) Building 410 Bolling AFB, DC 20332-6448		
8a. NAME OF FUNDING / SPONSORING ORGANIZATION AFOSR			8b. OFFICE SYMBOL (If applicable) NM		
9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-88-0152			10. SOURCE OF FUNDING NUMBERS		
8c. ADDRESS (City, State, and ZIP Code) Building 410 Bolling AFB, DC 20332-6448			PROGRAM ELEMENT NO. 61102F		TASK NO. 2304
11. TITLE (Include Security Classification) PARALLEL LOGIC PROGRAMMING AND PARALLEL SYSTEMS SOFTWARE AND HARDWARE			WORK UNIT A7		WORK UNIT ACCESSION NO.
12. PERSONAL AUTHOR(S) Professor Jack Minker					
13a. TYPE OF REPORT Final		13b. TIME COVERED FROM 1 Apr 88 TO 31 Mar 89		14. DATE OF REPORT (Year, Month, Day)	
15. PAGE COUNT					
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP			
19. ABSTRACT (Continue on reverse if necessary and identify by block number) <p>This report summarizes work performed under AFOSR-88-0152 on parallel logic programming, problem solving, and deductive databases. A parallel problem solving system, PRISM (Parallel Inference System), that was implemented on McMOB was ported to the BBN Butterfly machine. Two versions of PRISM were developed and are operational on the Butterfly: a message passing ring structure system and a shared-memory system.</p> <p>Experimental testing of PRISM on McMOB continued, while experiments were also conducted on the Butterfly systems. Three enhancements were made and completed during the grant period. These are: a capability to handle negated queries and a capability to assert and retract statements.</p> <p>In addition to the above, work continued in the area of information answers to queries in deductive databases. A thesis was completed on the subject. An interpreter was developed and is running, that can take restricted natural</p>					
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input type="checkbox"/> UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL DR ABRAHAM WAKSMAN			22b. TELEPHONE (Include Area Code) (202) 767-5027		22c. OFFICE SYMBOL NM

UNCLASSIFIED

language as input and can respond with a cooperative natural language output. In the area of parallel software development, the following were accomplished. theoretical work on slicing/splicing was completed. Tools were provided for software development using artificial intelligence techniques. AI software for massively parallel architectures was started.

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

UNIVERSITY OF MARYLAND
DEPARTMENT OF COMPUTER SCIENCE
COLLEGE PARK, MARYLAND
20742

AFOSR-TR- 89-1309

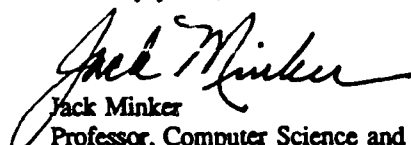
Telephone (301) 454-6119
July 28, 1989

Carey M. Fountain, Major, USAF
Department of the Air Force
Air Force Office of Scientific Research
Bolling Air Force Base
Washington, DC 20332-6448

Dear Major Fountain:

Enclosed is the Final Technical Report for Grant No. AFOSR-88-0152 of which I am the Principal Investigator. I greatly appreciate the support that has been given to me to permit this research to be accomplished. If there is additional information that you desire, please do not hesitate to contact me.

Sincerely yours,


Jack Minker
Professor, Computer Science and
Institute for Advanced Computer Studies

Accepted
July 29, 89

PARALLEL LOGIC PROGRAMMING
and
PARALLEL SYSTEMS SOFTWARE AND HARDWARE

FINAL PROGRESS REPORT
AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
GRANT NUMBER: AFOSR-88-0152

by

PROFESSOR JACK MINKER
Department of Computer Science
and
Institute for Advanced Computer Studies
University of Maryland
College Park, Maryland 20742

Jack Minker
Principal Investigator

Abstract

This progress report summarizes work performed under AFOSR-88-0152 on parallel logic programming, problem solving, and deductive databases. A parallel problem solving system, PRISM (Parallel Inference System), that was implemented on McMOB was ported to the BBN Butterfly machine. Two versions of PRISM were developed and are operational on the Butterfly: a message passing ring structure system and a shared-memory system.

Experimental testing of PRISM on McMOB continued, while experiments were also conducted on the Butterfly systems. Three enhancements were made and completed during the grant period. These are: a capability to handle negated queries and a capability to assert and retract statements.

In addition to the above, work continued in the area of informative answers to queries in deductive databases. A thesis was completed on the subject. An interpreter was developed and is running, that can take restricted natural language as input and can respond with a cooperative natural language output.

In the area of parallel software development, the following were accomplished. Theoretical work on slicing/splicing was completed. Tools were provided for software development using artificial intelligence techniques. AI software for massively parallel architectures was started.

1. Introduction

We describe research conducted in problem solving, deductive databases, and parallel systems software under Air Force Grant AFOSR-88-0152. A parallel problem solving system based on logic, PRISM (*PaRallel Inference System*), implemented on the McMOB parallel processor was ported to the BBN Butterfly machine. The ported PRISM system maintained the ring structure of the McMOB system. We shall refer to this system as PRISM-BMP. In addition, a shared-memory version of the system was developed for the Butterfly machine, which will be referred to as PRISM-SM. The PRISM-McMOB system will be referred to as PRISM. All three PRISM systems underwent experimental testing, and were enhanced in a number of ways.

In the following section we provide a description of the accomplishments under the grant. In the area of parallel problem solving, three PRISM systems are fully operational and tested on parallel architectures. Extensive experimentation was continued on PRISM and experimentation was started on the PRISM-BMP and PRISM-SM systems. An interpreter has been developed to implement our theory of informative answers for queries to deductive databases and problem solving systems. These and other accomplishments are described in Section 2.1.

In the area of parallel software development, we completed the research on slicing/splicing, developed tools for software development using techniques from Artificial Intelligence; and investigated AI software for massively parallel architectures. These accomplishments are described in Section 2.2.

As a consequence of the work accomplished during the past year, we have published:

- (1) 3 Ph.D. theses with partial support from the grant, [Gal88b, Koh87a, Sher87].
- (2) 1 Master of Science Degree Scholarly Paper, [Gaa88].
- (3) 1 journal article (invited), [Min88].
- (4) 2 journal articles accepted and to appear, [Cha88, Gra88]
- (5) 3 refereed conference papers, [Gal87a, HVM87, Lobo88].
- (6) 1 book chapter, [Gal88a].
- (7) 1 book chapter to appear (invited), [Gal88c].

Additional papers are in progress. A list of papers that have been written during the current grant is provided in Section 2.3. A total of 4 published papers have appeared in print, and a total of 3 additional papers have been accepted for publication during the grant period. The references in Section 4 of the proposal list all papers produced with support from the AFOSR during the several years that the Air Force has been supporting the research.

2. Accomplishments on Effort During Period October 15, 1987- September 1, 1988

This section is subdivided into three major parts. The first section, 2.1, describes the accomplished research with respect to PRISM - the parallel problem solving system. The second section, 2.2, describes the efforts for the development of parallel systems software and hardware for experimentation with parallel algorithms. Section 2.3 contains a list of all papers and reports written during the grant period.

Two major efforts were proposed:

- (1) Parallel Inference System Developments
- (2) Parallel Software/Hardware Developments

Each of these tasks and their sub-tasks are described in the following two sections.

2.1. Parallel Inference System Efforts

There were six major tasks that we planned to undertake during the proposed grant period. These are the first 6 tasks listed below. In addition, work was performed on the analysis of parallel algorithms, item 7.

- (1) *Perform Continued Experiments With PRISM on McMOB*
- (2) *Add Additional Features to PRISM*
- (3) *Integrate Kohli's Control Language Specification into PRISM*
- (4) *Incorporate the EDB into PRISM*
- (5) *Develop PRISM for the Butterfly and Connection Machines*
- (6) *Extend the Work on Informative and Cooperative Answers*
- (7) *Analysis of Parallel Algorithms*

Each of these topics is expanded upon in the following sections.

2.1.1. Perform Continued Experiments With PRISM on McMOB

Three classes of experiments were performed:

- (1) *Continued Experiments with the Current System.*

We have performed a large number of experiments on the current system that incorporate OR-parallelism. These experiments have been performed with problems that cover a wide range of domains including graph problems, combinatorial problems (the four queens problem), natural language processing, and others.

We have also developed a program that generates a logic program that conforms to the specification of an abstract AND/OR tree. That is, we can specify that a logic program be generated where the tree of the execution of the program has a particular shape. Each node can have a specified number of OR-branches, the solution(s) may be specified to be at a particular depth and there may be a specified number of AND-branches at a node. These programs were used to test the performance of the system on problems with given abstract properties. The purpose of these experiments was to determine the classes of problems for which the PRISM system is useful. Although we have started experimentation here, much more work remains to be done and the work will be continued.

Using programs collected from the literature and abstract benchmark programs we are able to obtain speed-ups of 50 factors using up to 80 processors. Abstract programs which are tailored to exploit parallelism at a very low cost have been shown to exploit parallelism almost perfectly (up to 75 factors of speed-up using 80 processors). Extrapolating from the data obtained we conclude that saturation for most of the problems considered would occur when using around 100 processors [Giu88]. A study was performed comparing the size of benchmark programs to the amount of parallelism obtained. The results indicate that increasing the size of a benchmark program by an order of magnitude only allows for a linear increase in the speed-up factor obtained.

Two issues involving the distribution of goals to processors were examined. First we examined techniques for preventing trivial goals from being distributed to remote processors. This was accomplished through the use of heuristics and program annotations. The PRISM control annotations allow the user to prevent unwanted OR-parallelism. In many cases a goal will unify with more than one clause. However, after simple tests, not expressible through unification, only one of the clauses will execute to completion. The annotations can prevent

these clauses from being executed in parallel and thus avoid the overhead associated with sending a goal to a remote processor. Heuristics were examined which prevent trivial goals from being sent to remote processors. The heuristics consider the number and types of literals in a goal and the size of terms in the goal. Although these heuristics are useful in many settings they are limited in that they only measure the size of a goal presuming it succeeds. The heuristics do not measure the possibility that a goal will fail immediately [Giu87].

A second issue in the distribution of goals to processors is communication protocols which allow processors to know when remote resources are available. This type of knowledge is needed for efficient performance. If no remote resources are available then a useless overhead results if an attempt is made to send queries to remote machines. If processors are available and not being exploited then parallelism is prevented. Communication schemes which tell processors when remote resources are available were examined and were shown to have a large affect on systems performance [Giu88].

(2) *Experiments with AND-Parallelism and Alternative Control Methods*

We have integrated AND-parallelism into the PRISM system. Because of other tasks, we have been unable to experiment with the two modes of AND-parallelism: independent and dependent.

In independent AND-parallelism the subproblems at a node do not share variables. Hence bindings that are returned do not have to be tested. In independent AND-parallelism there is no problem in sending the subproblems to another machine to be solved. The major problem is that one must not send trivial problems to be solved. In this case, the strategies for sending OR-branches to a new problem solver are applicable and can be used.

In dependent AND-parallelism complications arise in sending a problem to a new machine, since the bindings that are returned have to be sent to the parent machine in which the split was performed to test whether or not the bindings are compatible. Hence, additional strategies have to be devised and experimented with.

(3) *Experiments with the Constraint Machine*

Although the constraint machine is operational and integrated with the basic system, we have not performed extensive experiments with the system to determine the circumstances in which constraints will help. Such issues as the number of constraint machines vs. the number of problem solving machines are important to study. At the present time there can be only one constraint machine. We are planning to modify the system to develop the capability of having multiple constraint machines.

We have determined several areas in which integrity constraints can be useful to prune search spaces including type reasoning, database applications, and some generate and test programs. Constraints are used to detect failure without having to execute the goal. In some of the examples considered the constraints could, with some effort, be incorporated into the original logic program. However, this would result in code whose declarative meaning is obscure. By using a separate integrity constraint machine we have a mechanism to detect failure early while maintaining a clean declarative programming style.

The application areas identified show that constraints are useful if there is no overhead in checking for constraint violation. We plan to conduct experiments to determine how valuable constraints are in a parallel system. The use of constraints will increase the amount of messages passed between processors. It may turn out that the benefits associated with pruning the search space will be offset by the added communication between processors.

2.1.2. Add Additional Features to PRISM

There are three non-logical features that we are in the process of adding to the PRISM system. These are:

- (a) *Assert* a clause.
- (b) *Retract* a clause.
- (c) *No: Operator*.

These three capabilities will be useful in writing application programs in artificial intelligence. The *assert* capability will permit us to add lemmas derived by the system. This capability was needed when we wrote a meta-interpreter to simulate the behavior of a person performing real-time reasoning [MPS86]. The capability has been designed and implemented. The ability to *retract* data is also needed in other work that is being performed at Maryland by Don Perlis on a memory model [Per87]. We would like to test his memory model on

the PRISM system. Work on simulation of real-time reasoning and on the memory model was performed under a different grant. The retract capability has also been designed and implemented.

We have also designed and implemented the *not operator*. It will permit us to obtain negative answers to a query by negation as failure. The negation operator may be used only in the case where the literal is fully grounded and can operate only sequentially or in restricted AND-parallelism.

2.1.3. Integrate Kohli's Control Language Specification into PRISM

With partial support from this grant, a thesis has been written by Kohli [Koh87] that describes a compiler that permits a user to develop an interpreter with a control capability specified for a particular application. The compiler has been written and experiments have been conducted with the compiler output. An interpreter with the PROLOG control strategy was implemented and compared with PROLOG. Tests run on the VAX machine indicate that the compiled control operates approximately half as fast as PROLOG. However, for control structures that have to be implemented with a meta-interpreter on PROLOG, the compiled interpreter operates approximately ten times as fast as the meta-interpreter. Hence, the approach is both significant and viable for obtaining interpreters with control strategies different than that incorporated in PROLOG.

2.1.4. Incorporate the EDB into PRISM

Although the Extensional Database (EDB) had been implemented and tested with the simulated belt, it was not integrated and tested with the full system. At present EDB machines contain only ground, function-free atomic formulae, permit multiple relations to be stored in a single machine, and allow a single relation to reside in one or more machines. The EDB has been fully integrated into the McMOB system. A Master's Degree Scholarly Paper, supported by the grant, describes the EDB system [Gaa88].

2.1.5. Develop PRISM for the BBN Butterfly Machine and Connection Machine

We have implemented the PRISM system on the Butterfly architecture. Our objective was to obtain comparative analyses of the same system on two different architectures. In addition, the Butterfly machine has 128 nodes, while the McMOB system has only 16 nodes.

The PRISM system has been implemented using the C language. The C language is also available on the Butterfly. Hence, many of the programs needed only to be recompiled to be able to be run on the Butterfly. There are two major programs that have been developed. These are:

- (a) A loader that takes a specified configuration of PRISM machines (PSMs, IDBs, EDBs, and CSMs) and maps each machine onto a different node of the Butterfly.
- (b) A message passing system which allows PRISM machines to communicate. Four addressing modes were implemented to permit PRISM messages to be sent to alternative PRISM machines. The four capabilities are:
 - direct address: the message contains the specific address of the machine to which the message is being sent;
 - single pattern: the first machine which has the specified pattern is to pick up the message;
 - all: all machines receive the same message; and
 - all patterns: the message is sent to all machines with the specified pattern.

In the above, we translated PRISM to the Butterfly without taking advantage of the Butterfly shared-memory system. We refer to the PRISM system implemented on the Butterfly in the *message passing* mode as PRISM-BMP, while the PRISM system on McMOB is referred to as PRISM. Some experiments have been run to compare PRISM against PRISM-BMP. The experiments indicate that the systems have very similar parallel performance. PRISM-BMP is generally faster when using a single machine and becomes slower as more machines are used [Giu88].

In addition we have redesigned and implemented PRISM to take better advantage of the shared-memory of the Butterfly. The implementation, which only supports OR-parallelism, correctly implements the PRISM control annotations. In the implementation the proof tree is shared by all processors. A global work queue is used to distribute work to alternative processors. Initial results indicate that the system has a faster uni-processor speed than the message passing versions of PRISM [Dur88].

Although we did not anticipate implementing the new design, we were able to accomplish this task. The PRISM shared-memory system will be referred to henceforth as PRISM-SM. Because we implemented PRISM-SM, some of the other tasks that we had anticipated accomplishing may not be completed during the remainder of the grant period. Some experiments on the PRISM-SM system were performed and comparisons with the other two designs were conducted.

2.1.6. Extend the Work On Intelligent and Cooperative Answers

Work on generating cooperative answers to database queries was continued. As anticipated, a Ph.D. thesis has been written on this topic [Gal88b]. The thesis, done entirely at the University of Maryland with partial support from the grant, was awarded to Annie Gal at the University of Rennes, where it was defended in December 1988. She received her degree with High Honors.

Detailed heuristics concerning the incorporation of natural language into the informative answer process have been developed. We also implemented an interpreter that takes queries as input and provides informative answers to a user in natural language, as output. The interpreter builds on the meta-interpreter developed by Lobo and Minker [Lobo88]. We have augmented the meta-interpreter to permit the cooperative answer process to access the integrity constraints that have been employed. The use of integrity constraints is the means by which we are able to provide generality, and to achieve cooperative answers. In addition, ten heuristic rules have been developed to permit the most relevant of the many possible cooperative responses, to be output to the user [Gal87, Gal88a, Gal88b, Gal88c]. We believe that this general approach to providing cooperative answers to queries will be fundamental to expert systems and deductive databases.

To develop user-friendly systems, it is necessary to enhance the cooperative character of natural language dialogue between a user and a database. This research demonstrates how semantics already present in a database, in the form of *integrity constraints*, allow a database interface to respond more cooperatively to a questioner. This study suggests a new use for integrity constraints which have served mainly to control updates in databases, and to semantically optimize queries.

The domain-independent general approach to provide informative and cooperative answers to users can be summarized as follows. All information relevant to a user's query and to the enhancement of a cooperative answer must be collected. Misconceptions between a user's knowledge and a database are detected during this stage. A collection process has been developed where integrity constraints, the user's current query, and the database itself are consulted. A cooperative response, however, entails more than simply providing all information collected about a query. Heuristics have been specified to determine which information collected by the system should be submitted to a user. The rules developed are the major contribution of the research. They control the quality and quantity of information to be returned to a questioner. These rules may be summarized briefly as follows:

1. *Screening out misconceptions:*

When several misconceptions are discovered in a query, a selection takes place to choose those misconceptions which most likely correct the questioner's misunderstanding.

2. *Clarity of the resulting response:*

Under appropriate circumstances, a summary response, short and non-enumerative, may be more desirable (because it is clearer and less likely to mislead the user by generating false explanations), than a long enumerative answer.

3. *Relevance of the response:*

When an integrity constraint is related to a query, two further rules decide if the constraint is informative enough to be part of the cooperative response. For example, an integrity constraint which restricts the generality of a query is selected, while an integrity constraint which does not change the query in any way is not considered sufficiently informative to be explained to a questioner.

4. *Terseness of the Response:*

Sometimes explanations individually selected as part of a cooperative response are partially or totally redundant when brought together. Additional rules are used to choose the most appropriate explanation.

A detailed description of the method and heuristic rules used may be found in [Gal87,Gal88]. An implementation of the approach in the form of a natural language database interface has been designed and is currently under development.

The approach to generating cooperative responses is domain-independent, although it uses domain-specific information (contained in the form of integrity constraints, the user's query, and the database itself). The method utilizes knowledge already present in a database and can be applied to any deductive database (and also any relational database, since deductive databases are generalizations of relational databases). Logic provides a uniform language for all phases of the work, from the description of this approach to the implementation of the natural language interface.

2.1.7. Analyze Parallel Algorithms and Alternative Architectures

delim \$\$
gsize 11p

The thesis written by Sherlekar, [She87], applies graph separator theorems to solve problems in graph embeddings, VLSI layouts, and (sequential and parallel) algorithms for problems such as propositional satisfiability and trihedral scene recognition in computer vision. An $f(n)$ - separator theorem provides an efficient method to partition an n -vertex graph into two roughly equal parts by removing $O(f(n))$ vertices or edges. Good separator theorems (i.e., those for which $f(n) = \Theta(n)$) are known to exist for several classes of graphs such as planar, series-parallel, outerplanar, trees, and graphs of bounded genus. Separators have shown promise in formulating divide-and-conquer strategies for efficient solutions to several graph theoretic problems such as VLSI layouts, sparse Gaussian elimination, shortest paths, and network flow. The thesis shows how to use these techniques to solve the following problems embedding graphs into binary trees; layouts for separable graphs of arbitrary degree; planar layouts; Layouts for k -outerplanar graphs; and Separator-theoretic paradigms for designing algorithms.

2.2. Parallel Software Developments

There were three areas of work in the development of parallel software undertaken under the current grant. These were:

- (a) Completion of Work on Slicing/Splicing
- (b) Tools for Software Development using Artificial Intelligence Techniques
- (c) AI Software for Massively Parallel Architectures

2.2.1. Slicing/Splicing

This year has seen the completion of our involvement with the slicing/splicing work. Work at the beginning of this grant period focused on the conclusion of our theoretical analysis of these techniques [Wei87, Wei88]. The theoretical analysis demonstrated the clear potential for the development of a practicable slicing/splicing compiler for parallel computer systems. This work is currently underway at Xerox Palo Alto Research Center under the direction of Dr. Weiser.

2.2.2. Tools for Software Development Using Artificial Intelligence Techniques

The language ADA¹, developed by the DOD for delivery of computer software, is designed in part to promote the ability to design parallel programs by maximizing the re-use of software components. Unfortunately, the language specification addresses issues in the design and implementation of both reusable software modules and parallel programs, but not the identification and location of appropriate modules during the software development task. The goal of our research has been to approach this issue with a concentration on the use of symbolic information and artificial intelligence techniques. This work thus forms an attack on a critical, but as yet unsolved, problem in the software design process.

In the past year we have implemented a prototype system which demonstrates that AI technology combined with a knowledge base of software components can impact this problem. Our prototype AI-based reuse tool, called AIRS (Artificial Intelligence Reuse System) [HVM87], takes a description of program functionality and returns some suggested Ada packages for implementation of the objects described. An interaction is then begun in which the user can query for more information on these packages and thus discover other functional information which can be used later in the design and implementation process. The Ada modules handled in the prototype version consist of the EVB Grace components, a set of over 250 generic packages defining data structures.

AIRS uses a knowledge-based inference system which contains information about the operations, data types, and data structures contained in the packages. AIRS is designed to help in the task of choosing a representation and implementing it. This is done with a focus on reuse, rather than on algorithm synthesis or the like. The system takes the specification and finds appropriate generic packages to suggest for implementation.

A prototype of the AIRS system has been completed meeting the goals of the above design. The system is implemented in Common-Lisp and is currently working on a number of different machines. The system contains a knowledge base containing information about the Grace package structures, and can be used to find reusable components. We are presently developing a front-end interface for the system to make it more accessible.

2.2.3. Artificial Intelligence Software for Massively Parallel Architectures

Our research is directed towards the design and use of symbolic AI computing languages and techniques for the Connection Machine. Although the design of that machine was partially inspired by Scott Fahlman's NETL work, and although much interest in parallelism has been expressed by the AI community, few significant massively parallel AI systems have been designed. Those systems which have been built have generally centered on the fast numeric computation of distributed (connectionist) networks, or on special purpose software designed to handle one specific problem. Little work has been centered on the capabilities that massive parallelism can bring to standard AI tasks.

The goal of our work has been to bring some standard AI representational tools, particularly those of semantic network and frame systems, to the Connection Machine and examine what role the rapid computation of formerly inefficient algorithms can yield. We have designed, and begun implementing, a knowledge representation system which performs the efficient computations of inheritance intersections (i.e. "Find all instances which inherit from the vehicle class, have or inherit capabilities including X&Y and can be inferred to be in locations accessible from position A or B") [Hen88, Hen88a].

2.3. Papers and Reports Written

Below we list the papers or reports written during the grant period. As a consequence of the work, we have published 3 Ph.D. theses with partial support from the grant, [Gal88b, Koh87a, Sher87]; 1 Master of Science Degree Scholarly Paper, [Gaa88]; 1 journal article (invited), [Min88]; 2 journal articles accepted and to appear, [Cha88, Gra88]; 3 refereed conference papers, [Gal87, HVM87, Lobo88]; 1 book chapter, [Gal88a]; 1 book chapter to appear (invited), [Gal88c].

A list of all papers and reports written since the AFOSR first sponsored this work is given in the bibliography in Section 4. The list of papers and reports that have been published during the present grant period are:

¹ Ada is a Registered trademark of the US Gov't (AJPO).

- (1) [Cha88] Chakravarthy, U., Grant, J. and Minker, J. "Logic Based Approach to Semantic Query Optimization," *ACM TODS*, to appear
- (2) [Dur88] Durand, I., Giuliano, M. and Minker, J. "Implementations of PRISM on the Butterfly Multiprocessor," In Preparation, University of Maryland.
- (3) [Gaa88] Gaasterland, T. "Distributed Extensional Databases in PRISM," Scholarly Paper, Department of Computer Science, University of Maryland, College Park, MD. 20742, May 1988.
- (4) [Gal88a] Gal, A. and Minker, J. "Informative and Cooperative Answers in Databases Using Integrity Constraints", In: *Natural Language Understanding Logic Programming*, (V. Dahl and P. Saint-Dizier, Eds.), North Holland Publications, 1988, pp 277-300.
- (5) [Gal87a] Gal, A. and Minker, J. "Greater Cooperation between Database and User: Integrity Constraints Provide an Answer," *Proceedings First Annual Conference on Natural Language and Logic Programming*, Vancouver, Canada, 1987.
- (6) [Gal88b] Gal, A. "Cooperative Responses in Deductive Databases" *Thesis*, University of Maryland Technical Report, UMIACS-TR-88-55, CS-TR-2075, July 1988.
- (7) [Gal88c] Gal, A. and Minker, J. "Producing Cooperative Answers in Deductive Databases" In: *Logic and Logic Grammars for Language Processing* (P. Saint-Dizier and S. Szpakowicz, Eds.) L.S. Harvard Ltd. (to appear)
- (8) [Giu87] Giuliano, M., Kohli, M., Minker, J., Rajasekar, A., & Sherlekar, D. *Parallel Logic Programming in PRISM: Initial Experimental Work* Technical Report CS-TR-1887, Computer Science Department, University of Maryland.
- (9) [Giu88] Giuliano, M.E. Durand, I., Kohli, M., Minker, J. and Rajasekar, A. "Experiments with OR-Parallel Logic Programs on PRISM" In Preparation, University of Maryland.
- (10) [Gra88] Grant, J. and Minker, J. "Deductive Database Theories," *Knowledge Engineering Reviews*, invited paper (to appear).
- (11) [Hen87] Henderson, P. and Weiser, M., "The Visiprolog Environment," to appear in *IEEE Software*, 1987
- (12) *[Hen88] Hendler, J.A. *Integrating Marker-Passing and Problem-Solving: A spreading activation approach to improved choice in planning* Lawrence Erlbaum Associates, New Jersey, (In press).
- (13) *[Hen88a] Hendler, J. and Israel, B. "A Highly Parallel Implementation of a Marker-Passing Algorithm" Department of Computer Science, University of Maryland, TR CS-TR-2089, August, 1988.
- (14) [HVM87] Hendler, J., Vinciguerra, A. & Mogilensky, J. *An AI-Based Ada Reuse Tool* to be presented at the Third Annual Conference on Artificial Intelligence and Ada, Fairfax, Va., October, 1987.
- (15) [Koh87a] Kohli, M. "Controlling the Execution of Logic Programs" *Thesis* University of Maryland Technical Report, UMIACS-TR-87-39, CS-TR-1898, 1987.
- (16) [Koh87b] Kohli, M. and Minker, J. "Specifying Control for Logic Programs" *TR - 1935* University of Maryland, Oct. 1987.
- (17) [Lobo88] Lobo, J. and Minker, J. *A Metaprogramming Approach to Semantically Optimize Queries in Deductive Databases*, *Proceedings of the Second International Conference on Expert Database Systems*, (L. Kerschberg, Ed.), 1988, 387-420.
- (18) [Maz87] Mazurek, M. *Towards the Automatic Parallelization of Sequential Programs*, i. Scholarly Paper, Computer Science Department, University of Maryland, 1987. ii. With M. Weiser, Submitted *Acta Informatica*
- (19) [Min88] Minker, J. "Perspectives in Deductive Databases," *Journal of Logic Programming*, 1988:5:33-60.
- (20) [She87] Sherlekar D. "Graph Dissection Techniques for VLSI and Algorithms" *Thesis*, University of Maryland Technical Report, UMIACS-TR-87-40, CS-TR-1909, Sept. 1987.
- (21) [Wei88] Weiser, M. and Badger, L. "Automatic Detection and Use of Process Parallelism" *Proc. International Conference on Parallel Processing*, August, 1988.
- (22) [Wei87] Weiser, M. "Source Code" *IEEE Computer*, November, 1987.

3. Bibliography and References

Papers or reports, written with support from the AFOSR since the inception of the grant over several years, are preceded by an asterisk (*) References not listed with as asterisk are cited in the text.

- (1) [Bra78] Brachman, R.J. *A Structural Paradigm For Representing Knowledge* TR # 3605, Bolt, Baranek, And Newman, Ma. May, 1978.
- (2) *[Cao82] Cao, D., "Design of the Intensional Database System of the ZMOB Parallel Problem Solver", Technical Report, Computer Science Department, University of Maryland, 1982.
- (3) *[Cha85] Chakravarthy, U.S., "Semantic Query Optimization", Ph. D. Thesis, Department of Computer Science, University of Maryland, 1985.
- (4) *[Cha84] Chakravarthy, U.S., Fishman, D., and Minker, J., "Semantic Query Optimization in Expert Systems and Database Systems", University of Maryland, July 1984.
- (5) *[Cha82] Chakravarthy, U.S., Minker, J. and Tran, D., "Interfacing Predicate Logic Languages and Relational Databases". *Proceedings of the First International Logic Programming Conference*, September 14-17, 1982, Faculte des Sciences de Luminy Marseille, France, 91-98.
- (6) *[Cha88] Chakravarthy, U., Grant, J. and Minker, J. "Logic Based Approach to Semantic Query Optimization," *ACM TODS*, to appear
- (7) [C78] Clark, K.L., "Negation as Failure", In: *Logic and Data Bases*, (H. Gallaire and J. Minker, Eds), Plenum Press, 1978, pp. 293-322.
- (8) *[Dur88] Durand, I., Giuliano, M. and Minker, J. "Implementations of PRISM on the Butterfly Multiprocessor," In Preperation, Univiersity of Maryland.
- (9) *[Eis82] Eisinger, N., Kasif, S., and Minker, J., "Logic Programming: A Parallel Approach", *Proceedings of the First International Logic Programming Conference*, September 14-17, 1982, September 14-17, 1982, Faculte des Sciences de Luminy Marseille, France, 71-77.
- (10) *[Eis81] Eisinger, N., Kasif, S., and Minker, J., "Logic Programming: A Parallel Approach". Technical Report TR-1124, Computer Science Department, University of Maryland, December 1981.
- (11) [Fah79] Fahlman, S.E. *NETL: A System for Representing and Using Real World Knowledge*, MIT Press, Massachusetts, 1979.
- (12) *[Futo84] Futo, I., "A Constraint Machine to Control Parallel Search on Prism", Department of Computer Science, University of Maryland, 1984.
- (13) *[Gaa88] Gaasterland, T. "Distributed Extensional Databases in PRISM," Scholarly Paper, Department of Computer Science, University of Maryland, College Park, MD. 20742, May 1988.
- (14) *[Gal85] Gal, A. and Minker, J. "A Natural Language Database Interface Giving Cooperative Answers", Department of Computer Science, University of Maryland, 1985.
- (15) *[Gal88a] Gal, A. and Minker, J. "Informative and Cooperative Answers in Databases Using Integrity Constraints", In: *Natural Language Understanding Logic Programming*, (V. Dahl and P. Saint-Dizier, Eds.), North Holland Publications, 1988, pp. 277-300.
- (16) *[Gal87] Gal, A. and Minker, J. "Greater Cooperation between Database and User: Integrity Constraints Provide an Answer," *Proceedings First Annual Conference on Natural Language and Logic Programming*, Vancouver, Canada, 1987.
- (17) *[Gal88b] Gal, A. "Cooperative Responses in Deductive Databases" *Thesis*, University of Maryland Technical Report, UMIACS-TR-88-55, CS-TR-2075, July 1988.
- (18) *[Gal88c] Gal, A. and Minker, J. "Producing Cooperative Answers in Deductive Databases" In: *Logic and Logic Grammers for Language Processing* (P. Saint-Dizier and S. Szpakowicz, Eds.) *L.S. Harvard Ltd.* (to appear)
- (19) [GMN84a] Gallaire, H., Minker J. and Nicolas, J-M., "Logic and Databases: A Deductive Approach", *Computing Surveys*, Vol. 16, No. 2, pp. 153-185, June 1984.

- (20) [GMN84b] Gallaire, H., Minker, J. and Nicolas, J.-M., "Advances in Data Base Theory, Volume 2", Plenum Publishing Company, February, 1984.
- (21) *[Giu87] Giuliano, M., Kohli, M., Minker, J., Rajasekar, A., & Sherlekar, D. *Parallel Logic Programming in PRISM: Initial Experimental Work* Technical Report CS-TR-1887, Computer Science Department, University of Maryland.
- (22) *[Giu88] Giuliano, M.E. Durand, I., Kohli, M., Minker, J. and Rajasekar, A. "Experiments with OR-Parallel Logic Programs on PRISM" In Preparation, University of Maryland.
- (23) *[Gra88] Grant, J. and Minker, J. "Deductive Database Theories," *Knowledge Engineering Reviews*, invited paper (to appear).
- (24) *[Hen87] Henderson, P. and Weiser, M., "The Visiprog Environment," to appear in *IEEE Software*, 1987
- (25) *[Hen88] Hendler, J.A. *Integrating Marker-Passing and Problem-Solving: A spreading activation approach to improved choice in planning* Lawrence Erlbaum Associates, New Jersey, (In press).
- (26) *[Hen86] Hendler, J. and Reed, K. "SODA: The Software Designer's Aide" Technical Report (in preparation), Computer Science Department, University of Maryland, June, 1986.
- (27) *[Hen88a] Hendler, J. and Israel, B. "A Highly Parallel Implementation of a Marker-Passing Algorithm" Department of Computer Science, University of Maryland, TR CS-TR-2089, August, 1988.
- (28) *[HVM87] Hendler, J., Vinciguerra, A. & Mogilensky, J. *An AI-Based Ada Reuse Tool* to be presented at the Third Annual Conference on Artificial Intelligence and Ada, Fairfax, Va., October, 1987.
- (29) [Hil74] Hill, R., "LUSH Resolution and its Completeness", DCL Memo 78, Department of Artificial Intelligence, University of Edinburgh, August 1974.
- (30) [Hil85] Hillis, W.D *The Connection Machine*, MIT Press, Massachusetts, 1985.
- (31) *[Kas85a] Kasif, S. and Minker, J. "The Intelligent Channel: A Scheme for Result Sharing in Parallel Logic Programs", Proceedings International Joint Conference on Artificial Intelligence, August 1985.
- (32) *[Kas85b] Kasif, S., "Analysis of Parallelism in Logic Programs", Ph. D. Thesis, Department of Computer Science, University of Maryland, December 1985.
- (33) *[Kas83a] Kasif, S., Kohli, M., and Minker, J., PRISM: A Parallel Inference System for Problem Solving, Proceedings of the 8th International Joint Conference on Artificial Intelligence, 8-12 August 1983, Karlsruhe, West Germany, 544-546.
- (34) *[Kas83b] Kasif, S., Kohli, M., and Minker, J., "PRISM - A Parallel Inference System Based on Logic", Technical Report TR-1243, Computer Science Department, University of Maryland, February 1983.
- (35) *[Kas83c] Kasif, S., Kohli, M., and Minker, J., "PRISM: A Parallel Inference System for Problem Solving", Proceedings Logic Programming Workshops, Praiada Falesia, Algarve, Portugal, Universidade Nova De Lisboa, 26 June - 1 July 1983, 123-152.
- (36) *[Kas88] Kasif, S., Reif, J. and Sherlekar, D.D., "Formula Dissection: A Divide and Conquer Algorithm for Satisfiability," submitted for publication.
- (37) *[Koh86] Kohli, M. "Controlling the Execution of Logic Programs", Proposed Ph. D. Thesis, Department of Computer Science, University of Maryland, May 1986.
- (38) *[Koh83a] Kohli, M., and Minker J., "Control of Logic Programs Using Integrity Constraints" Proceedings Logic Programming Workshop 1983, 26 June - 1 July 1983, 123-152.
- (39) *[Koh83b] Kohli, M., and Minker, J., "A Theory of Intelligent Forward and Backward Tracking", Technical Report (in preparation), Computer Science Department, University of Maryland, March 1983.
- (40) *[Koh83c] Kohli, M., and Minker, J., "Intelligent Control Using Integrity Constraints", Proceedings of the National Conference on Artificial Intelligence, August 22-26, 1983, 202-205.
- (41) *[Koh87a] Kohli, M. "Controlling the Execution of Logic Programs" Thesis University of Maryland Technical Report, UMIACS-TR-87-39, CS-TR-1898, 1987.
- (42) *[Koh87b] Kohli, M. and Minker, J. "Specifying Control for Logic Programs" TR - 1935 University of Maryland, Oct. 1987.

- (43) *[Lobo88] Lobo, J. and Minker, J. *A Metaprogramming Approach to Semantically Optimize Queries in Deductive Databases*, *Proceedings of the Second International Conference on Expert Database Systems*, (L. Kerschberg, Ed.), 1988, 387-420.
- (44) *[Lyle84] Lyle, J. *Evaluating Variations on Program Slicing for Debugging*. Ph.D. Dissertation, Computer Science Dept., University of Maryland. December 1984.
- (45) *[Maz87] Mazurek, M. *Towards the Automatic Parallelization of Sequential Programs*, i. Scholarly Paper, Computer Science Department, University of Maryland, 1987. ii. With M. Weiser, Submitted *Acta Informatica*
- (46) *[Min88] Minker, J. "Perspectives in Deductive Databases," *Journal of Logic Programming*, 1988:5:33-60.
- (47) [MPS86] Minker, J., Perlis, D. and Subramanian, K., "A Parallel Self-Modifying Default Reasoning System", *Proceedings, AAAI-86*, Philadelphia.
- (48) *[Min85] Minker, J. Perlis, D., "Completeness Results for Circumscription", Department of Computer Science, University of Maryland, January 1985.
- (49) *[Min84] Minker, J. and Perlis D., "Applications of Protection of Circumscription", *Proceedings of the Conference on Automated Deduction* - , Napa, California, March 1984.
- (50) *[Min83a] Minker, J., and Perlis, D., "On The Semantics of Circumscription", Technical Report 1341, Computer Science Department, University of Maryland, August 1983.
- (51) *[Min82] Minker, J., et al., "Functional Description of the ZMOB Parallel Problem Solving System", Technical Note 1, Department of Computer Science, University of Maryland, December 1982.
- (52) *[Min83b] Minker, J., et al., "Parallel Problem Solving on ZMOB", *Proceedings of Trends and Applications 83*, Washington, D.C., 1983.
- (53) [Per87] Perlis, D., et al., "Life on a Desert Island", *Proceedings, Workshop on the Frame Problem*, April 1987, Lawrence, Kansas.
- (54) *[OTW85] O'Toole, J., Torek, C., and Weiser, M. Implementing XNS protocols for 4.2bsd. *Unix Users Conference*, Dallas TX. January 1985.
- (55) [Rous75] Roussel, P., "PROLOG: Manuel de Reference et d'Utilisation", Group d' Intelligence Artificielle, Universite d' Aix-Marseille, 1975.
- (56) [Sei85] Seitz, C.L. "The Cosmic Cube" *CACM* January, 1985.
- (57) *[She87] Sherlekar D. "Graph Dissection Techniques for VLSI and Algorithms" *Thesis*, University of Maryland Technical Report, UMIACS-TR-87-40, CS-TR-1909, Sept. 1987.
- (58) *[Wei88] Weiser, M. and Badger, L. "Automatic Detection and Use of Process Parallelism" *Proc. International Conference on Parallel Processing*, August, 1988.
- (59) *[Wei87a] Weiser, M. "Source Code" *IEEE Computer*, November, 1987.
- (60) *[Wei87b] Weiser, M. and Shneiderman, B., "Human Factors of Software Design and Development," in *Handbook of Human Factors*, ed. Gavriel Salvendy, John Wiley & Sons, 1987.
- (61) *[Wei85] Weiser, M., Kogge, S., McElvany, M., Pierson, R., Post, R., and Thareja, A. Status and Performance of the ZMOB Parallel Processing System. *IEEE CompCon conference*, San Francisco, CA, February 1985.

TABLE OF CONTENTS

Abstract	1
1. Introduction	1
2. Accomplishments on Effort During Period October 15, 1987- September 1, 1988	2
2.1. Parallel Inference System Efforts	2
2.1.1. Perform Continued Experiments With PRISM on McMOB	2
2.1.2. Add Additional Features to PRISM	3
2.1.3. Integrate Kohli's Control Language Specification into PRISM	4
2.1.4. Incorporate the EDB into PRISM	4
2.1.5. Develop PRISM for the BBN Butterfly Machine and Connection Machine	4
2.1.6. Extend the Work On Intelligent and Cooperative Answers	5
2.1.7. Analyze Parallel Algorithms and Alternative Architectures	6
2.2. Parallel Software Developments	6
2.2.1. Slicing/Splicing	6
2.2.2. Tools for Software Development Using Artificial Intelligence Techniques	7
2.2.3. Artificial Intelligence Software for Massively Parallel Architectures	7
2.3. Papers and Reports Written	7
3. Bibliography and References	9